ZEA, A Data Management Approach for SMR

Adam Manzanares
Co-Authors

• Western Digital Research
  – Cyril Guyot, Damien Le Moal, Zvonimir Bandic

• University of California, Santa Cruz
  – Noah Watkins, Carlos Maltzahn
Why SMR?

• HDDs are not going away
  – Exponential growth of data still exists
  – $/TB vs. Flash is still much lower
  – We want to continue this trend!

• Traditional Recording (PMR) is reaching scalability limits
  – SMR is a density enhancing technology being shipped right now.

• Future recording technologies may behave like SMR
  – Write constraint similarities
  – HAMR
Flavors of SMR

• SMR Constraint
  – Writes must be sequential to avoid data loss

• Drive Managed
  – Transparent to the user
  – Comes at the cost of predictability and additional drive resources

• Host Aware
  – Host is aware of SMR working model
  – If user does something “wrong” the drive will fix the problem internally

• Host Managed
  – Host is aware of SMR working model
  – If user does something “wrong” the drive will reject the IO request
SMR Drive Device Model

• New SCSI standard Zoned Block Commands (ZBC)
  – SATA equivalent ZAC

• Drive described by zones and their restrictions

<table>
<thead>
<tr>
<th>Type</th>
<th>Write Restriction</th>
<th>Intended Use</th>
<th>Con</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>None</td>
<td>In-place updates</td>
<td>Increased Resources</td>
<td>CZ</td>
</tr>
<tr>
<td>Sequential Preferred</td>
<td>None</td>
<td>Mostly sequential</td>
<td>Variable Performance</td>
<td>SPZ</td>
</tr>
<tr>
<td>Sequential Required</td>
<td>Sequential Write</td>
<td>Only sequential</td>
<td></td>
<td>SRZ</td>
</tr>
</tbody>
</table>

• Our user space library (libzbc) queries zone information from the drive
  – https://github.com/hgst/libzbc
Why Host Managed SMR?

• We are chasing predictability
  – Large systems with complex data flows
    • Demand latency windows that are relatively tight from storage devices
    • Translates into latency guarantees from the larger system

• Drive managed HDDs
  – When is GC triggered
  – How long does GC take

• Host Aware HDDs
  – Seem ok if you do the “right” thing
  – Degrade to drive managed when you don’t

• Host Managed
  – Complexity and variance of drive internalized schemes are now gone
Host Managed SMR seems great, but …

• All of these layers must be SMR compatible
  – Any IO reordering causes a problem
  – Must not happen at any point between the user and device

• What about my new SMR FS?
  – What is your GC policy?
  – Is it tunable?
  – Does your FS introduce latency at your discretion?

• What about my new KV that is SMR compatible?
  – See above
  – In addition, is it built on top of a file system?
What Did We Do?

- Introduce a zone and block based extent allocator [ZEA]
- Write Logical Extent [Zone Block Address]
  - Return Drive LBA
- Read Extent [Logical Block Address]
  - Return data if extent is valid
- Iterators over extent metadata
  - Allows one to build ZBA -> LBA mapping
ZEA Performance vs. Block Device

(a) Start Up Time

(b) Write Performance

(c) Read Performance
ZEA Is Just An Extent Allocator, What Next?

- **ZEA + LevelDB**
- **LevelDB** is KV store library that is widely used
  - LevelDB Backend API is good for SMR
    - Write File Append Only, Read Randomly, Create & Delete Files, Flat Directory
  - Lets Build a Simple FS compatible with LevelDB
ZEA + LevelDB Performance

(a) Write Time

(b) Sync Write Time

(c) Read Time
Lessons Learned

• **ZEA is a lightweight abstraction**
  – Hides sequential write constraint from application
  – Low overhead vs. a block device when extent size is reasonable
  – Provides addressing flexibility to application
    • Useful for GC

• **LevelDB integration opens up usability of Host Managed SMR HDD**

• **Unfortunately LevelDB not so great for large objects**
  – Ideal Use case for SMR drive would be large static blobs
What Is Left?

• What is a “good” interface above ZEA
• Garbage collection policies
  – When and How
• How to use multiple zones efficiently
  – Allocation
  – Garbage collection

• Thanks for listening